CLAIM AMENDMENTS

Please amend the claims as indicated:

1. (currently amended) ~~A computer system comprising~~ In a computer system that includes:

at least one hardware processor; and

a first operating system (COS) initially installed to run on the hardware processor at a most-privileged, system level, the system level being defined as an operational state with permission to directly access predetermined physical resources of the computer,

a method comprising:

initializing the computer with the COS;

~~the COS forming means for initializing the computer;~~

~~a kernel that forms a second operating system;~~

~~loading means for loading the kernel~~ via the COS, loading and ~~for~~ starting execution of ~~the~~ a kernel, the kernel thereupon substantially displacing the COS from the system level and itself running at the system level; and

~~the kernel forming means for~~ handling requests for system resources via the kernel, including scheduling execution of the COS on the hardware processor(s).

2. (original) A method as in claim 1, in which the step of loading the kernel comprises:

loading a load call module within the COS;

upon initialization of the computer, calling a loading module from the load call module, whereupon the loading module loads the kernel.

3. (original) A method as defined in claim 2, in which the step of loading the load call module within the COS comprises installing the load call module as a driver within the COS.

4. (currently amended) In a computer system that includes:

at least one hardware processor that has a hardware instruction pointer; and

a first operating system (COS) initially installed to run on the hardware processor at a most-privileged, system level, the system level being defined as an operational state with permission to directly access predetermined physical resources of the computer;

a method comprising:

initializing the computer with the COS;

loading a load call module within the COS;

upon initialization of the computer, calling a loading module from the load call module, whereupon the loading module loads a kernel, which forms a second operating system;

via the COS starting execution of the kernel, the kernel thereupon substantially displacing the COS from the system level and itself running at the system level;

~~A method as in claim 2, in which the computer includes at least one processor, which has a hardware instruction pointer, and~~ the step of loading the kernel including ~~the following sub-steps~~:

via the loading module, setting the hardware instruction pointer and forwarding of interrupts and faults generated by the processor and by predetermined ones of the system resources to point into a memory address space allocated to and controlled by the kernel, the kernel thereby handling requests for system resources.

5. (original) A method as in claim 4, further including the following steps:

after initialization of the computer, transferring from the COS to the kernel a list of devices initially controlled by the COS, the devices being included among the system resources; and

classifying the devices and control of the devices into the following groups (which may be empty):

host-managed devices, which are controlled by the COS;

reserved devices, which are controlled by the kernel;

and shared devices, which may be controlled by either the COS or the kernel.

6. (original) A method as defined in claim 5, further comprising including a mass storage controller as one of the shared devices.

7. (original) A method as defined in claim 6, in which the mass storage controller is a SCSI device.

8. (original) A method as defined in claim 5, further comprising including a network adapter as one of the shared devices.

9. (original) A method as defined in claim 5, further comprising the steps of forwarding interrupts generated by host-managed devices to the COS via the kernel, and handling such interrupts within the COS.

10. (original) A method as defined in claim 9, further including the step of delaying handling of interrupts that are forwarded to the COS and that are generated by host-managed devices until a subsequent instance of running of the COS.

11. (original) A method as defined in claim 10, further including the step, upon sensing, in the kernel, an interrupt raised by any host-managed device, of masking the interrupt until the subsequent instance of running of the COS, thereby avoiding multiple recurrences of the interrupt.

12. (original) A method as defined in claim 1, further including the step of installing at least one virtual machine (VM) to run on the kernel via a virtual machine monitor (VMM).

13. (currently amended) In a computer system that includes:

at least one hardware processor;

a first operating system (COS) initially installed to run on the hardware processor at a most-privileged, system level, the system level being defined as an operational state with permission to directly access predetermined physical resources of the computer;

a method comprising:

initializing the computer with the COS;

via the COS, loading and starting execution of a kernel, which forms a second operating system, the kernel thereupon substantially displacing the COS from the system level and itself running at the system level, the kernel thereby handling requests for system resources;

~~A method as defined in claim 12, further including the following steps:~~

installing at least one virtual machine (VM) to run on the kernel via a virtual machine monitor (VMM);

in the kernel, separately scheduling the execution of the COS and of each VM, the COS and the ~~VM's~~ VMs thereby forming separately schedulable and separately executing entities; and

within the kernel, representing each schedulable entity as a corresponding world, each world comprising a world memory region with a respective world address space and storing a respective world control thread.

14. (original) A method as defined in claim 13, further including the step of switching worlds, which step comprises:

under control of the kernel, storing current state data for a currently executing schedulable entity in a kernel-controlled memory region;

disabling exceptions;

loading state data for a subsequently executing schedulable entity;

starting execution of the subsequently executing schedulable entity; and

enabling exceptions.

15. (original) A method as defined in claim 14, in which the state data for each schedulable entity includes exception flags, memory segments, an instruction pointer, and descriptor tables, which are loaded into an exception flags register, memory segment registers, an instruction pointer register, and descriptor tables, respectively.

16. (original) A method as defined in claim 14, in which the computer includes a plurality of hardware processors, further including the following steps:

in the kernel, separately scheduling the execution of each processor, the processors thereby also forming separately schedulable entities;

within the kernel, representing each processor as a corresponding system world, each system having a respective system world address space and a respective system world control thread.

17. (original) A method as defined in claim 16, further including the step of allocating, for each processor, a separate memory mapping cache.

18. (original) A method as defined in claim 13, in which each VM includes a virtual processor, a virtual operating system (VOS), and an I/O driver, loaded within the VOS, for an I/O device, the method further comprising the following steps:

allocating a shared memory space that is addressable by both the kernel and the VM's I/O driver,

transferring an output set of data from the VM to the I/O device according to the following sub-steps:

via the VM's I/O driver, setting a pointer to the output set of data in the shared memory region and generating a request for transmission;

in the kernel, upon sensing the request for transmission:

retrieving the output set of data from a position in the shared memory region indicated by the pointer and transferring the retrieved output set of data to a physical transmit buffer portion of the shared memory region;

transferring the output data set from the physical transmit buffer portion to the I/O device;

transferring an input set of data from the I/O device to the VM according to the following sub-steps:

in the kernel,

copying the input set of data into a physical receive buffer portion of the shared memory region;

setting the pointer to the physical receive buffer portion;

issuing to the VMM an instruction to raise an interrupt;

in the VM, upon sensing the interrupt raised by the VMM, retrieving the input set of data from the physical receive buffer portion of the shared memory region indicated by the pointer;

whereby the input and output data sets may be transferred between the VM and the I/O device via only the kernel.

19. (original) A method as defined in claim 18, further comprising completing the sub-steps for transferring the output set of data upon sensing only a single request for transmission.

20. (original) A method as defined in claim 18, in which:

the I/O device is a network connection device for data transfer to and from a network; and

the input and output data sets are network packets.

21. (original) A method as defined in claim 12, further including the following steps:

mapping a kernel address space, within which the kernel is stored and is addressable by the kernel, into a VMM address space, within which the VMM is stored and which is addressable by the VMM.

22. (original) A method as defined in claim 21, in which the computer has a segmented memory architecture, the memory being addressable via segment registers, further including the step of setting a segment length for the VMM larger than a minimum length necessary to fully contain both the VMM and the kernel, whereby the step of mapping the kernel address space within the VMM address space may be performed free of any need to change a corresponding segment register.

23. (original) A method as defined in claim 12, in which each VM includes a virtual processor, a virtual operating system (VOS), and a virtual disk (VDISK), the method further including carrying out the following steps within the kernel:

partitioning the VDISK into VDISK blocks;

maintaining an array of VDISK block pointers, the array comprising a plurality of sets of VDISK block pointers;

maintaining a file descriptor table in which is stored file descriptors, each file descriptor storing block identification and allocation information, and at least one pointer block pointer;

each pointer block pointer pointing to one of the sets of VDISK block pointers; and

each VDISK block pointer identifying the location of a respective one of the VDISK blocks.

24. (original) A method as defined in claim 1, further including the following steps:

halting execution of the kernel;

reinstating a state of the first operating system that existed before the loading of the kernel; and

resuming execution of the first operating system at the most-privileged system level;

the kernel thereby being functionally removed from the computer.

25. (currently amended) In a computer system that includes:

at least one hardware processor, which has a hardware instruction pointer;

a first operating system (COS) initially installed to run on the hardware processor at a most-privileged, system level, the system level being defined as an operational state with permission to directly access predetermined physical resources of the computer;

a method comprising:

initializing the computer with the COS;

via the COS, loading and starting execution of a kernel, which forms a second operating system, the kernel thereupon substantially displacing the COS from the system level and itself running at the system level, the kernel thereby handling requests for system resources;

halting execution of the kernel;

reinstating a state of the first operating system that existed before the loading of the kernel; and

resuming execution of the first operating system at the most-privileged system level;

the kernel thereby being functionally removed from the computer;

~~A method as defined in claim 24,~~ in which:

~~A) the computer includes at least one processor, which has a hardware instruction pointer;~~

~~B~~ A) the step of loading the kernel includes the following sub-steps:

i) loading a load call module within the COS;

ii) upon initialization of the computer, calling a loading module from the load call module, whereupon the loading module loads the kernel;

iii) after initialization of the computer, transferring from the COS to the kernel a list of devices initially controlled by the COS; and

iv) classifying the devices and control of the devices into the following groups (which may be empty):

a) host-managed devices, which are controlled by the COS;

b) reserved devices, which are controlled by the kernel; and

c) shared devices, which may be controlled by either the COS or the kernel;

v) via the loading module, setting the hardware instruction pointer and forwarding of interrupts and faults generated by the processor and by predetermined ones of the physical resources to point into a memory address space allocated to and controlled by the kernel;

~~C~~ B) the step of reinstating the state of the first operating system includes the following steps:

i) restoring interrupt and fault handling from the kernel to the first operating system;

ii) transferring control of host-managed and shared devices from the kernel to the first operating system; and

iii) removing the kernel from an address space of the first operating system.

26. (Original) A method for managing resources in a computer, which includes at least one processor that has a hardware instruction pointer, the method comprising the following steps:

A) initializing the computer using a first operating system (COS), the COS itself running at a most-privileged, system level, the system level being defined as an operational state with permission to directly access predetermined physical resources of the computer, the physical resources including physical devices;

B) loading a kernel via the COS, the kernel forming a second operating system, this step of loading the kernel comprising:

i) loading a load call module within the COS;

ii) upon initialization of the computer, calling a loading module from the load call module, whereupon the loading module loads the kernel;

iii) via the loading module, setting the hardware instruction pointer and forwarding interrupts and faults generated by the processor and by predetermined ones of the physical resources to point into a memory address space allocated to and controlled by the kernel;

C) starting execution of the kernel, the kernel thereupon substantially displacing the COS from the system level and itself running at the system level; and

D) submitting requests for system resources via the kernel;

E) after initialization of the computer, transferring from the COS to the kernel a list of the devices initially controlled by the COS; and

F) classifying the devices and control of the devices into the following groups (which may be empty):

i) host-managed devices, which are controlled by the COS;

ii) reserved devices, which are controlled by the kernel; and

iii) shared devices, which may be controlled by either the COS or the kernel; and

G) forwarding interrupts generated by host-managed devices to the COS via the kernel, and handling such interrupts within the COS.

27. (currently amended) A method for managing resources in a computer, which includes at least one processor that has a hardware instruction pointer, the method comprising the following steps:

A) initializing the computer using a first operating system (COS), the COS itself running at a most-privileged, system level, the system level being defined as an operational state with permission to directly access predetermined physical resources of the computer, the physical resources including physical devices;

B) loading a kernel via the COS, the kernel forming a second operating system, this step of loading the kernel comprising:

i) loading a load call module within the COS;

ii) upon initialization of the computer, calling a loading module from the load call module, whereupon the loading module loads the kernel;

iii) via the loading module, setting the hardware instruction pointer and forwarding interrupts and faults generated by the processor and by predetermined ones of the physical resources to point into a memory address space allocated to and controlled by the kernel;

C) starting execution of the kernel, the kernel thereupon substantially displacing the COS from the system level and itself running at the system level; and

D) submitting requests for system resources via the kernel;

E) after initialization of the computer, transferring from the COS to the kernel a list of the devices initially controlled by the COS; and

F) classifying the devices and control of the devices into the following groups (which may be empty):

i) host-managed devices, which are controlled by the COS;

ii) reserved devices, which are controlled by the kernel; and

iii) shared devices, which may be controlled by either the COS or the kernel; and

G) forwarding interrupts generated by host-managed devices to the COS via the kernel, and handling such interrupts within the COS;

H) installing at least one virtual machine (VM) to run on the kernel via a virtual machine monitor (VMM); and

I) in the kernel, separately scheduling the execution of the COS and of each VM, the COS and the ~~VM's~~ VMs thereby forming separately schedulable and separately executing entities.

28. (currently amended) A computer system comprising:

at least one hardware processor;

a first operating system (COS) initially installed to run on the hardware processor at a most-privileged, system level, the system level being defined as an operational state with permission to directly access predetermined physical resources of the computer, the COS forming means for initializing the computer;

a kernel ~~means~~ that forms a second operating system;

a ~~loading means~~ loader comprising computer-executable code for loading the kernel means via the COS and for starting execution of the kernel means, the kernel ~~means~~ thereupon substantially displacing the COS from the system level and itself running at the system level; and

the kernel ~~means is provided~~ including a software module for handling requests for system resources, including scheduling execution of the COS on the hardware processor(s).

29. (currently amended) A system as in claim 28, in which the ~~loading means~~ loader includes a loading driver installed within the COS.

30. (currently amended) A computer system comprising:

at least one hardware processor that has a hardware instruction pointer;

a first operating system (COS) initially installed to run on the hardware processor at a most-privileged, system level, the system level being defined as an operational state with permission to directly access predetermined physical resources of the computer, the COS initializing the computer;

a kernel that forms a second operating system;

a loader comprising computer-executable code for loading the kernel via the COS and for starting execution of the kernel, the kernel thereupon substantially displacing the COS from the system level and itself running at the system level;

the kernel including a software module for handling requests for system resources;

~~A system as in claim 28, in which:~~

~~the processor has a hardware instruction pointer;~~

the ~~loading means is~~ loader being further provided for setting the hardware instruction pointer and forwarding interrupts and faults generated by the processor and by predetermined ones of the system resources to point into a memory address space allocated to and controlled by the kernel ~~means~~.

31. (currently amended) A system as in claim 30, in which:

the system resources include devices initially controlled by the COS;

the ~~loading means~~ loader is further provided for transferring, after initialization of the computer, from the COS to the kernel ~~means~~ a list of the devices initially controlled by the COS and for classifying the devices and control of the devices into the following groups (which may be empty):

host-managed devices, which are controlled by the COS;

reserved devices, which are controlled by the kernel ~~means~~; and

shared devices, which may be controlled by either the COS or the kernel ~~means~~.

32. (original) A system as in claim 31, in which at least one of the shared devices is a mass storage controller.

33. (currently amended) A system as defined in claim 28, further comprising:

at least one virtual machine (VM); and

a virtual machine monitor (VMM);

in which the VM is installed to run on the kernel ~~means~~ via the VMM.

34. (currently amended) ~~A system as defined in claim 33,~~ A computer system comprising:

at least one hardware processor;

a first operating system (COS) initially installed to run on the hardware processor at a most-privileged, system level, the system level being defined as an operational state with permission to directly access predetermined physical resources of the computer, the COS forming means for initializing the computer;

a kernel that forms a second operating system;

a loader comprising computer-executable code for loading the kernel means via the COS and for starting execution of the kernel means, the kernel thereupon substantially displacing the COS from the system level and itself running at the system level;

a virtual machine monitor (VMM);

at least one virtual machine (VM) installed to run on the kernel via the VMM;

the kernel including software modules

for handling requests for system resources;

~~in which the kernel means is further provided:~~

for separately scheduling the execution of the COS and of each VM, the COS and the ~~VM's~~ VMs thereby forming separately schedulable and separately executing entities; and

for representing each schedulable entity as a corresponding world, each world comprising a world memory region with a respective world address space and storing a respective world control thread.

35. (currently amended) A system as defined in claim 34, in which the kernel ~~means~~ is further provided:

for storing current state data for a currently executing schedulable entity in a kernel-controlled memory region;

for disabling exceptions;

for loading state data for a subsequently executing schedulable entity;

for starting execution of the subsequently executing schedulable entity; and

for enabling exceptions;

the kernel ~~means~~ thereby being provided for switching worlds.

36. (original) A system as defined in claim 35, in which the state data for each schedulable entity includes exception flags, memory segments, an instruction pointer, and descriptor tables, which are loaded into an exception flags register, memory segment registers, an instruction pointer register, and descriptor tables, respectively.

37. (currently amended) A system as defined in claim 34, in which:

the computer includes a plurality of hardware processors;

the kernel ~~means~~ is further provided:

for separately scheduling the execution of each processor, the processors thereby also forming separately schedulable entities;

for representing each processor as a corresponding system world, each system having a respective system world address space and a respective system world control thread.

38. (original) A system as defined in claim 37, further comprising a separate memory mapping cache for each processor.

39. (currently amended) A computer system ~~as defined in claim 33, further~~ comprising:

at least one hardware processor;

a first operating system (COS) initially installed to run on the hardware processor at a most-privileged, system level, the system level being defined as an operational state with permission to directly access predetermined physical resources of the computer, the COS forming means for initializing the computer;

a kernel that forms a second operating system;

a loader comprising computer-executable code for loading the kernel means via the COS and for starting execution of the kernel means, the kernel thereupon substantially displacing the COS from the system level and itself running at the system level;

the kernel handling requests for system resources;

a virtual machine monitor (VMM);

at least one virtual machine (VM) installed to run on the kernel via the VMM;

within each VM, a virtual processor, a virtual operating system (VOS), and an I/O driver for an I/O device loaded within the VOS;

a shared memory space that is addressable by both the kernel ~~means~~ and the VM's I/O driver, the shared memory space storing input data and output data for transfer between the VM and the I/O device;

in which:

the VM's I/O driver ~~forms means~~ comprises computer-executable code for setting a pointer to output data in the shared memory region and generating a request for transmission;

the kernel ~~means~~ is further provided, upon sensing the request for transmission:

for retrieving the output data from a position in the shared memory region indicated by the pointer and transferring the retrieved output data to a physical transmit buffer portion of the shared memory region;

for outputting the output data from the physical transmit buffer portion to the I/O device;

for receiving the input data from the I/O device;

for copying the input data into a physical receive buffer portion of the shared memory region;

for setting the pointer to the physical receive buffer portion;

for issuing to the VMM an instruction to raise an interrupt;

the VM's I/O driver ~~forming means~~ is further provided, upon sensing the interrupt raised by the VMM, for retrieving the input data from the physical receive buffer portion of the shared memory region indicated by the pointer;

whereby the input and output data may be transferred between the VM and the I/O device via only the kernel ~~means~~.

40. (original) A system as defined in claim 39, in which:

the I/O device is a network connection device for data transfer to and from a network; and

the input and output data are network packets.

41. (currently amended) A system as defined in claim 33, further comprising:

a kernel address memory portion, within which the kernel ~~means~~ is stored and is addressable by the kernel ~~means~~;

~~means~~ a mapping module comprising computer-executable code for mapping the kernel address memory portion into a VMM address space, within which the VMM is stored and which is addressable by the VMM.

42. (currently amended) A system as defined in claim 41, in which:

the computer has a segmented memory architecture;

segment registers via which the memory is addressed; and

a segment length for the VMM that is larger than a minimum length necessary to fully contain both the VMM and the kernel ~~means~~.

43. (currently amended) A system as defined in claim 33, in which:

each VM includes a virtual processor, a virtual operating system (VOS), and a virtual disk (VDISK);

the VDISK is partitioning into VDISK blocks;

the kernel ~~means~~ is further provided:

for maintaining an array of VDISK block pointers, the array comprising a plurality of sets of VDISK block pointers;

for maintaining a file descriptor table in which is stored file descriptors, each file descriptor storing block identification and allocation information, and at least one pointer block pointer;

each pointer block pointer pointing to one of the sets of VDISK block pointers; and

each VDISK block pointer identifying the location of a respective one of the VDISK block.

44. (original) A computer system comprising:

at least one hardware processor;

a first operating system (COS) initially installed to run on the hardware processor at a most-privileged, system level, the system level being defined as an operational state with permission to directly access predetermined physical resources of the computer, the COS forming means for initializing the computer;

a kernel means that forms a second operating system;

loading means for loading the kernel via the COS and for starting execution of the kernel, the kernel thereupon substantially displacing the COS from the system level and itself running at the system level;

the kernel means is provided for handling requests for system resources;

in which:

the processor has a hardware instruction pointer;

the loading means is further provided for setting the hardware instruction pointer and forwarding interrupts and faults generated by the processor and by predetermined ones of the system resources to point into a memory address space allocated to and controlled by the kernel means;

the system resources include devices initially controlled by the COS;

the loading means is further provided for transferring, after initialization of the computer, from the COS to the kernel means a list of the devices initially controlled by the COS and for classifying the devices and control of the devices into the following groups (which may be empty):

host-managed devices, which are controlled by the COS;

reserved devices, which are controlled by the kernel means; and

shared devices, which may be controlled by either the COS or the kernel means.

45. (currently amended) A computer system comprising:

at least one hardware processor;

a first operating system (COS) initially installed to run on the hardware processor at a most-privileged, system level, the system level being defined as an operational state with permission to directly access predetermined system resources of the computer, the COS forming means for initializing the computer;

a kernel means that forms a second operating system;

loading means for loading the kernel means via the COS and for starting execution of the kernel means, the kernel means thereupon substantially displacing the COS from the system level and itself running at the system level;

at least one virtual machine (VM); and

a virtual machine monitor (VMM);

in which:

the VM is installed to run on the kernel means via the VMM;

the kernel means is provided for handling requests for system resources;

the processor has a hardware instruction pointer;

the loading means is further provided for setting the hardware instruction pointer and forwarding interrupts and faults generated by the processor and by predetermined ones of the system resources to point into a memory address space allocated to and controlled by the kernel means;

the system resources include devices initially controlled by the COS;

the loading means is further provided for transferring, after initialization of the computer, from the COS to the kernel means a list of the devices initially controlled by the COS and for classifying the devices and control of the devices into the following groups (which may be empty):

host-managed devices, which are controlled by the COS;

reserved devices, which are controlled by the kernel means; and

shared devices, which may be controlled by either the COS or the kernel means;

for separately scheduling the execution of the COS and of each VM, the COS and the ~~VM's~~ VMs thereby forming separately schedulable and separately executing entities; and

the kernel means is further provided:

for representing each schedulable entity as a corresponding world, each world comprising a world memory region with a respective world address space and storing a respective world control thread;

for storing current state data for a currently executing schedulable entity in a kernel means-controlled memory region;

for disabling exceptions;

for loading state data for a subsequently executing schedulable entity;

for starting execution of the subsequently executing schedulable entity; and

for enabling exceptions;

the kernel means thereby being provided for switching worlds.